Terminological Importation for Adapting Reusable Knowledge Representation Components in the KSM Environment

José L. Sierra, Martín Molina

Department of Artificial Intelligence
Technical University of Madrid
Campus de Montegancedo s/n
28660 Boadilla del Monte, Madrid, SPAIN
Tel: 34-91-3367390, FAX: 34-91-3524819

{jlsierra,mmolina}@dia.fi.upm.es
htpp://www.dia.fi.upm.es/

Abstract.

This paper describes the adaptation approach of reusable knowledge representation components used in the KSM environment for the formulation and operationalisation of structured knowledge models. Reusable knowledge representation components in KSM are called primitives of representation. A primitive of representation provides: (1) a knowledge representation formalism (2) a set of tasks that use this knowledge together with several problem-solving methods to carry out these tasks (3) a knowledge acquisition module that provides different services to acquire and validate this knowledge (4) an abstract terminology about the linguistic categories included in the representation language associated to the primitive. Primitives of representation usually are domain independent. A primitive of representation can be adapted to support knowledge in a given domain by importing concepts from this domain. The paper describes how this activity can be carried out by mean of a terminological importation. Informally, a terminological importation partially populates an abstract terminology with concepts taken from a given domain. The information provided by the importation can be used by the acquisition and validation facilities to constraint the classes of knowledge that can be described using the representation formalism according to the domain knowledge. KSM provides the LINK-S language to specify terminological importation from a domain terminology to an abstract one. These terminologies are described in KSM by mean of the CONCEL language. Terminological importation is used to adapt reusable primitives of representation in order to increase the usability degree of such components in these domains. In addition, two primitives of representation can share a common vocabulary by importing common domain CONCEL terminologies (conceptual vocabularies). It is a necessary condition to make possible the interoperability between different, heterogeneous knowledge representation components in the framework of complex knowledge - based architectures.

1. Introduction

The use of software components has gained an increasing interest in the last years. The modern programming environments gives the possibility of building a great part of the application by selecting, configuring and assembling reusable, pre-existing building blocks. The KSM environment is a knowledge modelling and knowledge – based system construction framework that includes the idea of an extensible library of reusable knowledge representation components to give support to the operationalisation of knowledge models. The basic assumptions of the KSM approach to formulate and to operationalise structured knowledge models can be summarised in the following points:

- Models are structured in terms of bodies of expertise called *knowledge areas*. A knowledge area can be decomposed by mean of other knowledge areas.
- The leaves of this hierarchy of knowledge areas (the so – called *primary knowledge areas*) can be associated to basic representation components called *primitives of representation*. Primitives of representation provide the mechanisms for encoding the domain knowledge associated to each primary area and the basic inference tasks to be carried out using this knowledge.
- Primary knowledge areas can share a basic domain terminology, which is described in terms of *conceptual vocabularies*, to ensure the consistency between their knowledge.

A primitive of representation can be understood as a reusable knowledge representation component, which is selected from a library, in order to provide a symbolic representation for operationalising a piece of domain knowledge. This component includes a declarative way of describing the knowledge associated to a given knowledge body. Simultaneously a primitive of representation can provide a set of inference

mechanisms optimised for the formalism. From this viewpoint, to build structured knowledge models in the KSM framework can be largely viewed as assembling and configuring reusable components in a component oriented programming environment.

The use of local, unrelated modules to capture the different bodies of knowledge in a given domain requires some mechanisms to ensure the consistence between the terminology used in each module, yet preserving the domain independent nature of the primitives of representation. The main aim of this paper is to show how this consistence can be achieved by using the conceptual vocabularies associated to the primary knowledge areas. When a primitive of representation is selected to support the knowledge of a primary knowledge area, it is supposed to *import* some of the conceptual vocabularies associated with the primary area. The paper establishes the idea of *conceptual vocabulary importation* as a *terminological importation* process: concepts from the terminologies associated to the different primary areas involved in the model are *imported* into the *abstract terminology* associated to the linguistic concepts managed by the primitive of representation. So, the paper is structured as follows. Section two is an overview of the KSM environment for the formulation and operationalisation of structured knowledge models. Section three describes the idea of a primitive of representation as reusable, domain – independent, knowledge representation component. Section four introduces the CONCEL language used in KSM to describe terminologies. Section five describes the theory of adapting primitives of representation by mean of an importation of conceptual vocabularies. Section six describes the LINK-S language, which provides the mechanisms to express the importation of terminologies, used in KSM to associate primitives of representation to primary areas. Section seven drafts an example illustrating the ideas introduced in the paper. Section eight compares terminological importation with related approaches. Section nine describes some conclusions and future work.

2. The KSM environment

KSM (Knowledge Structured Manager) [Cuena,Molina,94; Cuena,Molina,97] is a software environment supporting an approach for the formulation and operationalisation of structured knowledge models. To formulate a knowledge model in KSM three main perspectives are defined:

- The knowledge area perspective, which plays the role of central structure of the model as a structured collection of knowledge bodies.
- The task perspective, that describes the problem - solving behaviour of the model.
- The vocabulary perspective, which includes the basic terms shared by several knowledge models.

The knowledge - area perspective is used for presenting a general image of the model where each module represents what is called *knowledge area*. In general, a knowledge area identifies a body of expertise that explains a certain problem - solving behaviour of an intelligent agent. Typically, a knowledge area is associated to a professional skill, a qualification or speciality of an expert. For instance, in a medical domain there could be knowledge areas such as infectious diseases, therapies, heart diseases, etc. The whole knowledge model is a hierarchical structure of knowledge areas in such way that there is a top - level area representing the entire model. This area is divided (using the *part - of* relation) into other more detailed sub-areas that, in their turn, are divided into other simpler areas and so on, developing the whole hierarchy. A bottom level area is called *primary knowledge area* and corresponds to an elementary module that may be operationalised by using basic software building blocks.

Knowledge areas are not passive modules but they provide different services represented by a set of tasks. The task perspective presents a functional description for each task using a tree of task-method-subtasks. A *task* is a goal than can be achieved by knowing about a certain knowledge area. The task receives a set of input data and generates a set of output data as a reasoning result. Examples of tasks are: medical diagnosis of a patient, assignment of a set of offices to a group of people, design of the machinery of an elevator and mineral classification. The *method* describes how to carry out the task by using a particular problem - solving strategy. Examples of methods are: establish-and-refine, propose-and-revise, generate-and-test, and heuristic classification. In KSM methods are formulated using a particular language called LINK [Molina et al, 98]. This language allows the description of the lines of reasoning of problem solving methods with two main parts: the data flow section and the control flow section (where production rules are used to establish the execution order of sub-tasks).

Finally, the vocabulary perspective is formulated by mean of a set of components called conceptual vocabularies. A *conceptual vocabulary* defines a basic terminology used by several knowledge areas. A

vocabulary defines a partial view including the basic terms that are common to different knowledge bases. In KSM vocabularies are formulated using a particular description language called CONCEL, which will be described in a subsequent section in this paper.

The structure of knowledge-areas, task and vocabularies is called *generic* model, given that it is general and reusable. To develop a model for a particular domain the developer creates a quasi-isomorphic structure of knowledge areas specialised in the domain as an instantiation of the general description. For each generic knowledge area there will be one or more domain knowledge-areas, following the same relation established by the generic model. The developer particularises the domain structure writing particular knowledge bases, creates domain conceptual vocabularies and she may also redefine at the domain level that generic control knowledge defined in methods using the LINK language.

The previous model is *implementation - independent* (i.e it must be operationalised by using computational constructs that produce the executable version on the computer). In order to do so, it is necessary either to translate the model into an executable version by applying methods in software engineering, or to use high level reusable knowledge representation components that implement basic problem - solving techniques. In KSM the second solution is adopted by using the concept of *primitive of representation*. This kind of components can be associated to primary knowledge areas in order to give support to the representation of their knowledge together with the basic inference processes using this knowledge. This association requires an *adaptation* step as it is discussed in the next section.

3. The primitive of representation and its adaptation

The KSM environment provides an extensible library of primitives of representation to provide the means of describing the domain knowledge associated to the primary knowledge areas together with a set of inference tasks which can be associated with the tasks in the primary areas. The concept of primitive of representation as a reusable building block to built knowledge – based and conventional applications is detailed in [Molina et al, 97].

Briefly, a primitive of representation can be viewed as a pair <L, T>, where L is a representation language and $T \equiv \{T_i\}$ is a set of inference tasks that can be performed with the sentences written in the language L. For instance, a possible primitive of representation can be a *rule based* primitive, with a language to represent production rules, and a task *deduction* to perform inference in a given set of rules. Furthermore each task T_i is described in terms of a set $I \equiv \{I_i\}$ of *inputs roles*, and a set $O \equiv \{O_i\}$ of *output roles*. Each input role describes the dynamic information required for the task to achieve its inference goal, and the outputs roles are a description of the information generated as the result of such a inference. For instance, *deduction* in the *rule based* primitive could have as input roles a set of *initial facts* to be considered in the inference process, and a set of *goals* which would be proved or refuted using the initial facts and the set of rules written in the primitive's language. As output role this task could have the set of *derived facts*.

To achieve inferences, a set $M_i \equiv \{M_{ij}\}$ of *inference methods* must be associated to each task T_i. For instance, two possible methods could be associated to the *deduction* task: *forward chaining* and *backward chaining*. In addition to the solution for their associated tasks, the inference methods are supposed to support explanatory capabilities about their solutions. Such capabilities require an abstract description of the dynamic behaviour of the method in order to provide understandable explanations. It is not needed a complete description of such behaviour (the complete description is the implementation of the method!). Instead, a set of abstract milestone states of the dynamic can be provides under the assumption that the explanation is given in terms of instances of such states.

The conceptual structure of a primitive of representation is sketched in Figure 1. The elements of such description are the following:

- A module oriented to give support to knowledge acquisition and validation in terms of the representation language. Such a module could include several acquisition methods such as one based on direct knowledge editing, several types of machine learning, modules providing translation from reusable ontologies, etc.

- A set of tasks and the associated inference methods, as described previously.

- A set of internal information structures that give support to the different activities of the primitive. Such activities includes recording internal representations of the sentences written in the languages, storing internal results produced by the execution of the inference methods and supporting the creation of contexts oriented to support the non deterministic execution of the inference methods.

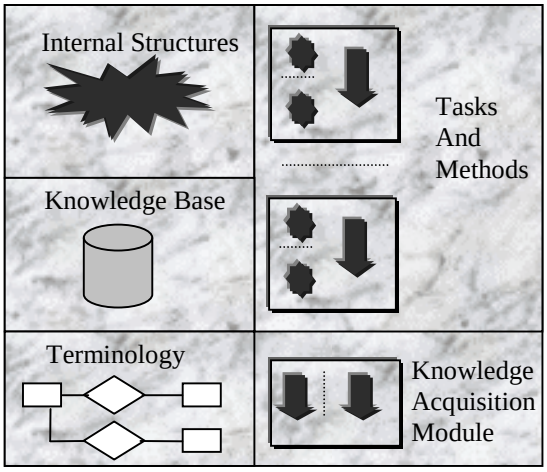


Figure 1. Structure of a primitive of representation. The main parts of the primitive are: a knowledge representation formalism, a set of tasks and inference methods, a knowledge acquisition module with several acquisition and validation services, a set of internal structures to give computational support to the primitive, and an abstract terminology about the linguistic categories of the representation formalism.

- A description of the *abstract terminology* related to the representation language supplied with the primitive. This terminology is involved with the linguistic categories included in the language. For example, a language about production rules could include linguistic categories about concepts used in a rule precondition or a rule action, attributes of these concepts, and associated domains for their values. This abstract terminology could be viewed as a set of knowledge roles to be filled with domain dependent terms. If the rule - based primitive was applied in the formulation of knowledge about rules relating symptoms with diseases in a clinical domain, the abstract *concepts* could be related with specific symptoms and diseases, and the abstract *attributes* could be associated to domain features of these symptoms and diseases. Knowledge acquisition and validation facilities could use these associations to constrain the feasible values used in the formulation of the rules. Furthermore, a commitment about the vocabulary used by several domain independent, isolated primitives can be obtained defining a proper association between their abstract terminologies and a shared domain one. This commitment is necessary to provide interoperability between these components, as sketched in Figure 2.

The explicit incorporation of the abstract terminology containing the linguistic categories of the representation language is the starting point to adapt the component in a given domain. For instance, a primitive oriented to the representation of clinical rules could be obtained from a generic rule based primitive establishing a right association with the appropriate domain vocabulary. On the other hand, the adaptation of the primitive for representing knowledge bodies in other domains can be achieved using different associations. For instance, the *rule based* primitive can be adapted to represent rules about the identification of situations in a traffic domain associating traffic detectors to rule precondition concepts, detector measures to rule precondition attributes, and traffic situations to rule post-conditions concepts. Similarly, a *frame- based* primitive with an abstract terminology in terms of frame names, slots, slot attributes and slot domains could be adapted in the clinical domain or in the control traffic domain. In both domains this primitive could give support to represent associations of control policies to problematic situations. In the clinical domain, it could be accomplished by associating frame names with diseases, slots with medicines, and attributes with administration intervals and amount of medicine to be submitted to the patient. In the traffic domain, frame names could be mapped into traffic situations, slots into control devices, and attributes into the state of such devices. Because diseases in the clinical diagnose rules are

the same that diseases in the therapy patterns, and traffic situations in the traffic identification rules are the same that the situations associated to the frames in the control plans, there is an agreement between the vocabularies used by both primitives. This agreement is a necessary condition to achieve the interoperability of both components. The next sections describe how these ideas can be carried out into the KSM framework.

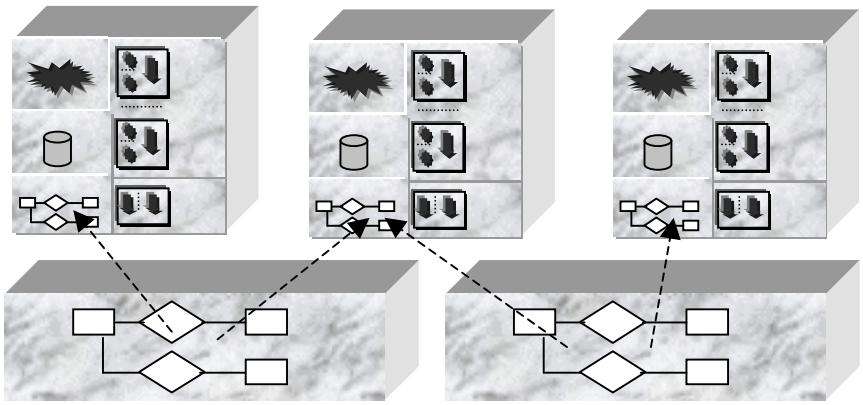


Figure 2. The association of concrete terminologies with abstract ones establish the common vocabulary between primitives. This terminological agreement is a necessary requirement to be met in order to achieve the interoperability between the knowledge representation components.

4. The CONCEL language for the description of conceptual vocabularies

The association of abstract terminologies and specific ones requires, first at all, an appropriate formalism to describe such terminologies. For this purpose, KSM provides with a special language called CONCEL. The main CONCEL construct is the so – called *concept*. A concept can be either a class of entities in the universe of discourse (a *concept class*) or a specific entity (a *concept instance*). Concept instances are derived from a concept class by mean of the `instance of` construct. In addition, a concept class can be derived from a (unique) super-class using the `subclass of` construct. There is a root of the class hierarchy called `concept`. A class can not be a (direct or indirect) super-class of itself. Duplicate names of concepts are forbidden as well.

Concepts have associated *attributes*. Attributes associated to classes have associated a *domain* and, optionally, a default value. The domains in CONCEL can be *any,boolean*, *numeric*, a numerical *interval*, an *enumeration* of strings or *instance of C*, where C is the name of a concept class. A subclass can redefine an attribute if the associated domain *entails* the domain in the redefinition according to the following rules:

- All domain *d* entails itself.
- *Any* entails to all domain *d.*
- *Numeric* entails all numerical intervals.
- A numerical interval entails all its subintervals.
- An enumeration entails all its sub-enumerations.
- An *instance of C* domain entails all domains of the form *instance of S*, where S is a subclass of C.

The default values can be redefined elsewhere. Default values and values in instances can be either single values and set - valued values (in both case the consistence between values and domains is mandatory). Figure 3 shows the syntax of the CONCEL language. Figure 4 shows a (very simplified) CONCEL definition related with general terminology in a traffic domain.

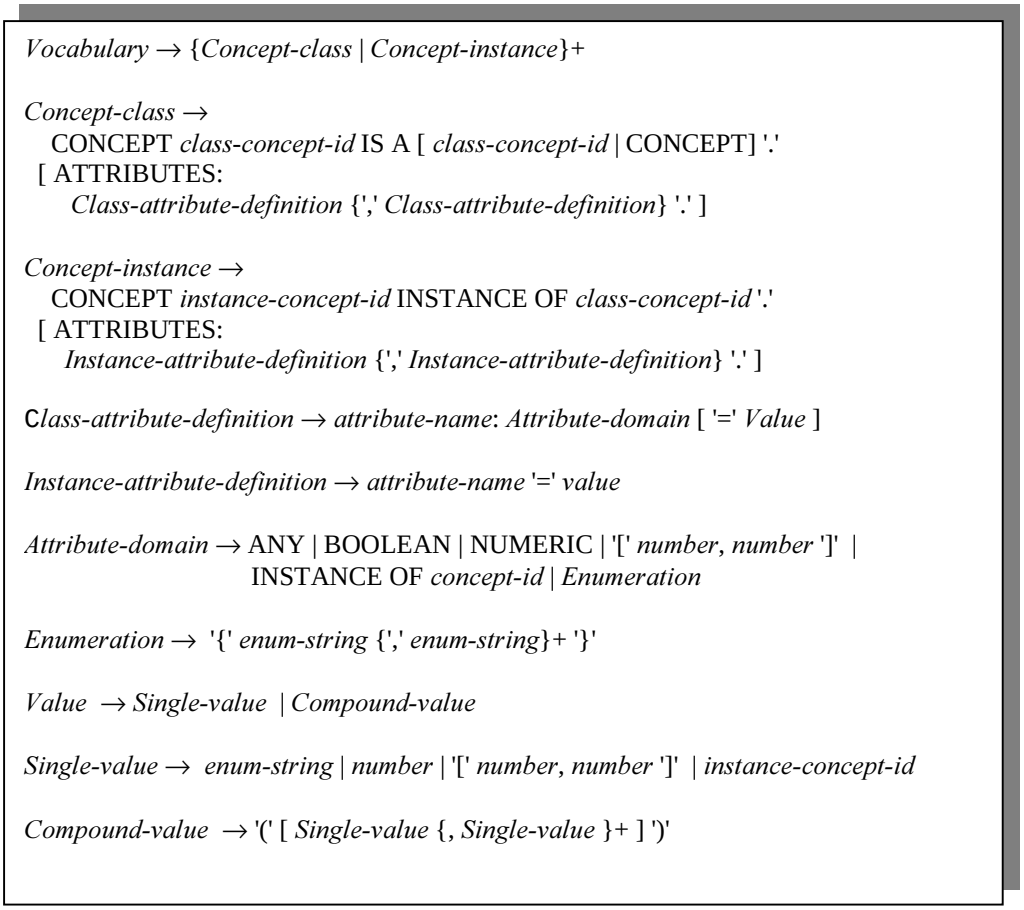


Figure 3. CONCEL syntax. The notation used is EBNF-like. | denotes alternatives, [ ] optional constructions, { } 0 - ∞ iteration, and { }+ 1 - ∞ iteration. Non terminal symbols are written in *Cursive* (with the first letter capitalised), classes of terminal symbols in *cursive* (with the first letter no capitalised), and particular terminal symbols enclosed into ' '.

The CONCEL semantic is derived from the intended meaning of the syntax in a straightforward manner. Firstly the attributes associated with a concept, and the values associated with these attributes are defined. The attributes and values associated with an instance I are defined as:

(1) For each clause $a = v$ in the definition of I, I has associated the attribute a with the value v.
(2) Let C the class of I. If a is associated with C, being v its value in the association, and it is not possible to associate a with I according to (1), a is associated to I with value v.

The attributes associated to a class and the values of the association are defined as follow:

(1) There are not attributes associated with `concept`.
(2) Let C a class. If in its definition there is a clause of the type $a:d=v$, a is associated to C with v as value.
(3) Let C a class and C' its super-class. If a is associated to C' with value v and it is not possible to associate a to C according to (2), a is associated to C with value v.
(4) Let C a class. If in its definitions there is a clause of the type $a:d$ and it is not possible to associate a to C according to (2) or (3), a is associated to C with value $\perp$, where $\perp$ denotes an *unknown* value.

Then, the denotations of values, attributes and concepts are defined according to a universe of discourse U, given by a set of objects:

- $\perp$, numbers and enumeration strings are interpreted as themselves.
- A concept instance I denotes an object in U.
- Attributes associated with a concept instance are denoted by relations between the object associated to the instance and the denotation of the value (or set of values) associated with the attribute.
- A concept class is the set that consists of the objects denoted by their instances and the union of the sets associated with their direct subclasses.

CONCEL definitions are useful to describe terminologies in an object - oriented style. These terminologies constitute the basic vocabulary shared by different knowledge bodies. These conceptual

vocabularies can be associated with the primary areas in a knowledge model structured according to the KSM approach. Then, suitable primitives of representation must be associated with these primary areas to formalise their knowledge. Primitives of representation are supposed to *import* the conceptual vocabularies associated to the primary knowledge areas. The next section formally states such *terminological importation.*

```
CONCEPT Road IS A Concept.
ATTRIBUTES:
   Sections: INSTANCE OF Section = ( ),
   Possible situations: INSTANCE OF Traffic Situation = ( ),
   Control devices: INSTANCE OF Control devices = ( ).

CONCEPT Section IS A Concept.
ATTRIBUTES:
   Previous Sections: INSTANCE OF Section = ( ),
   Next Sections: INSTANCE OF Section = ( ),
   Traffic Detector: INSTANCE OF Detector.

CONCEPT Detector IS A Concept.
ATTRIBUTES:
   Intensity Measure: NUMERIC,
   Occupancy Measure: NUMERIC.

CONCEPT Speed Detector IS A Detector.
ATTRIBUTES:
   Speed Measure: NUMERIC.

CONCEPT Traffic Situation IS A Concept.
ATTRIBUTES:
   Presence: {null, medium, high}.

CONCEPT Control Device IS A Concept.
ATTRIBUTES:
   State: ANY.

CONCEPT Ramp Semaphore IS A Control Device.
ATTRIBUTES:
    State: {close, medium, open}.
```

Figure 4. A simplified CONCEL description about traffic. A *section* is a transversal cut in a road. A *detector* is a device that takes different measures about the traffic traversing a section. These measures are *intensity* (the amount of cars traversing the detector in a time interval), *occupancy* (the fraction that a given detector is activated by cars in a time interval) and *speed* (only *speed detectors* are capable to provide this measure). A *control device* is one that can be used to control the traffic behaviour. Control devices can be in a given *state*. A kind of *control devices* is a ramp semaphore, which can be used to control the access of cars to an entry point in a highway.

5. Terminological importation

The basic abstract vocabulary postulated by the primitive is given by its abstract terminology. Such a vocabulary can be represented in CONCEL terms. The knowledge acquisition and validation module of the primitive is supposed to provide such abstract conceptual vocabulary, and to accept an *association* of the abstract conceptual vocabulary to a specific one: it is, the primitive representation is supposed to *import* a concrete conceptual vocabulary. Such importation activity is intended as the definition of abstract instances from the instances in a concrete terminology, and will be named *terminological importation.*

Informally, a terminological importation partially *populates* an abstract terminology with instances taken from concrete one. This population activity supposes to define, for each imported instance, how their attributes holds in the abstract terminology and the meaning of the concrete values in terms of the abstract ones. For this purpose it is sufficient to specify the meaning of the enumerate concrete values in the abstract vocabulary. For instance, Figure 6 shows how the abstract terminology associated to the *rule based* primitive (Figure 5) can be populated importing concepts from the concrete traffic terminology

given in the Figure 4. It is stated that the detector instances could be precondition concepts, and the traffic situation could be postcondition concepts. In addition, the measure attributes associated with the detectors can be used as attributes in the preconditions of the rules, and the presence of a traffic situation can be used as attribute in a rule postcondition. This idea can be stated formally as follows:

A *terminological importation* from a (concrete) terminology D to an (abstract) terminology A is a tuple $\Phi \equiv <\sigma,\phi,\varphi>$ where:

- σ is a partial function from concepts in D to classes in A.
- φ is a partial function from values in enumerations of D to values in enumerations of A.
- ϕ is a partial function from attributes in D to attributes in A, such as, if $\phi(a)$ is defined: (1) if a is associated with c $\phi(a)$ is associated with $\sigma(c)$, and (2) the domain of $\phi(a)$ entails the domain of a transformed by φ.

```
CONCEPT Rule Concept IS A Concept.
ATTRIBUTES:
   Attribute: ANY.

CONCEPT Precondition Concept IS A Rule Concept.

CONCEPT Postcondition Concept IS A Rule Concept.
```

Figure 5. The abstract terminology associated to the *rule based* primitive of representation. Rules are built from concepts with attributes. There are concepts associated to rule preconditions and concepts associated to rule post-conditions.

```
Detector MAYBE A Precondition Concept.

      Detector.Intensity Measure MAYBE A Precondition Concept.Attribute.
      Detector.Occupancy Measure MAYBE A Precondition Concept.Attribute.
      Speed Detector.Speed Measure MAYBE A Precondition Concept.Attribute.

Traffic Situation MAYBE A Postcondition Concept.
      Traffic Situation.Presence MAYBE A Postcondition Concept.Attribute
```

Figure 6. Specification of a terminological importation from the traffic terminology to the *rule based* primitive of representation terminology. The concepts in the rule precondition may be detectors, and the allowed attributes those associated to the detector measures. The post-conditions may be qualified traffic situations.

In this way, the terminological importation could be thought as augmenting the abstract terminology with new definitions derived from the concrete one. In this way, the application of the importation sketched in Figure 6 could generate a new terminology with *Detector* defined as a subclass of *Precondition Concept* and *Traffic Situation* as a subclass of *Postcondition Concept*. However the operation provides additional second – order information about the attributes (recorded in the ϕ partial function), because, for instance, the attribute *Presence* of *Traffic Situation* is associated with the attribute *Attribute* of *Postcondition Concept*. These second order dependencies can not directly be described in CONCEL.

The importation of a set of conceptual vocabularies for a given primitive supposes to establish a terminological importation from the terminology given by the union of the different conceptual vocabularies[1] to the abstract terminology associated with the primitive. The information given by the importation can be used for the acquisition and validation module in order to check the correctness of the acquired knowledge. For example, the validation procedures in the *rule based* primitive would be supposed to reject a rule deriving the measure in a detector. Similarly, the measures associated with the elements in the preconditions would be compatible with the numerical domain. In addition, this information can be used for the inference methods in order to check their applicability in terms of the

[1] The conflicts between names in the vocabularies are solved using as scope the name of the vocabulary.

underlying ontological assumptions about the inputs roles of the associated tasks. For instance, the inference methods could be presuming that the *initial facts* role is containing those facts about elements in the preconditions of the rules and, at the same time, not present in the post-conditions. It is possible to build the primitive of representation to make this type of assumptions *with independence of* the concrete importation: the validation procedures only assume that such a importation would be carried out *when the primitive will be adapted to a given domain.* Domain - independent knowledge representation components, which exploit domain dependent terminologies to contraint the classes of knowledge that can be described in their representation formalisms, can be written using this idea. Furthermore, different components can share the same vocabulary using the importation mechanism. This is a necessary condition to ensure the interoperability between such components. On the other hand, this mechanism can be efficiently implemented: it only needs an array linking abstract definitions with concrete ones. Next section presents the LINK – S language that operationalise the idea in the KSM environment.

6. The LINK – S language for associating primitives of representation to primary knowledge areas

The association of a primitive of representation to a primary knowledge area requires the following steps:

- To specify how the primitive of representation can import the conceptual vocabularies associated with the primary area. This specification supposes to specify a terminological importation from the concrete terminology associated with the union of these conceptual vocabularies to the abstract terminology related to the primitive of representation.
- To specify how the tasks associated to the primary area are carried out in terms of the tasks associated to the primitive of representation. The hypothesis here is that each task in the area can be performed by a task in the primitive. This hypothesis is reasonable because if the task had needed the assembling of several tasks, the primary area would not be a primary area anymore. In such a case the assembling process could be carried out writing a suitable LINK method describing as the task could be carried out.
- Once a task in the primitive is associated with the knowledge area, to specify which method would be selected to carry out such a task.

These steps are covered specifying the needed *metaknowledge* in a language called LINK– S. LINK- S is similar to LINK in the sense that it offers the needed *glue* to put several elements together. The main difference is that, while LINK allows the specification of control policies by mean of production rules, LINK-S is oriented to specify only static associations between knowledge categories.

LINK – S gives support for describing two different kinds of associations:

- Association of tasks in the primitive of representation to tasks in the primary areas.
- Association of concepts in the conceptual vocabularies to the abstract terminology provided by the primitive of representation.

The association of tasks is performed specifying *data flow expressions* from the inputs of the primary area's task to the inputs of the primitive's task, and expressions transforming the outputs of the primitive's task to the output roles of the primary area. These data flow expressions are analogous to those used in the LINK language (flow concatenation, flow selection, flow labelling, complex flow construction, etc) and are discussed with detail in [Molina et al, 98a]. The importation of concepts from the conceptual vocabularies are given by *maybe* sentences analogous to those used in the Figure 6. In order to solve potential ambiguities a *dot* operator, which specify the qualification of concept names with vocabulary names, attribute names with concept names and enumeration items with attribute names, is used. LINK – S syntax is specified in Figure 7.

The *vocabulary importation* section in a LINK – S specification serves, in this way, to describe the terminological importation from concrete terminologies. The σ,ϕ,φ partial functions associated with the importation are amalgamated in a set of *maybe* clauses describing the entries of these functions. Next section develops a complete example showing as LINK – S is used in order to associate primitives of representation to primary areas.

Association →
PRIMITIVE *primitive name* '.'
[*Terminological association*]
Task association

Terminological association → VOCABULARY IMPORTATION {*Importation*}

Importation → *Qualified Name* MAYBE A *Qualified Name* '.'

Qualified Name → *name* {'.' *name*}

Task association → TASK ASSOCIATIONS {*Task association*}+

Task association →
primary area task PERFORMED BY *primitive task* '(' *inference method* ')'
[INPUT *Role association* {, *Role association*}]
[OUTPUT *Role association* {,*Role association*}] '.'

Role association → *data flow expression* '->' *role*

Figure 7. LINK - S syntax. This language serves for specifying both associations from tasks in the primitive to tasks in the primary area and the specification of terminological importations from conceptual vocabularies to the abstract terminology of the primitive of representation.

7. Example

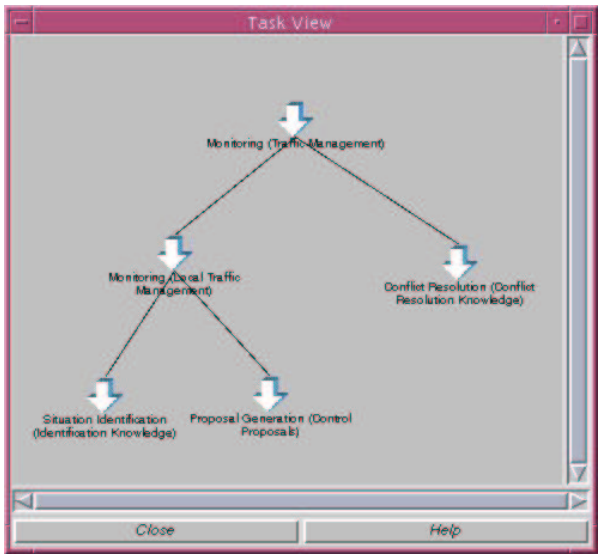
This section describes an example about the use of KSM to develop an application with reusable knowledge representation components that are adapted by mean of terminological importation. The problem to be solved is the construction of a system for detecting and monitoring anomalous situations in a traffic domain. The proposed system will be a simplified variant of the TRyS system that are been used for traffic control in several Spanish cities [Cuena et al,96]. The emphasis of the example is focused on the primitives of representation adaptation process.

Figure 8 shows a generic model formulated in terms of the knowledge area approach for traffic control. According to it the traffic control problem is formulated on different local areas in order to reduce complexity. In each area a local control proposal is generated by applying the following (shallow) decision procedure:

- Identifying the traffic situation in the area. The knowledge to do it is represented by the *identification* primary knowledge area in the model.
- Mapping the situation to a suitable control plan in terms of the control device states in the area. The knowledge to do it is represented by the *control proposals* primary knowledge area.

Once local proposals has been submitted a *conflict resolution* knowledge is applied that is oriented to solve the possible contradictions between these proposals. Contradictions arise because maybe two different proposals try to set two different states over a control device (*physical incompatibilities)*, or there is a nonsense assignation of control states to two different control devices (*logical incompatibilities*). Figure 9 shows the functional view of this model in terms of the tasks associated to each knowledge area.

Once such a model is formulated the common terminology associated with the different primary areas must be described in terms of one (or several) *generic* conceptual vocabularies. Such vocabularies are CONCEL vocabularies describing the concept classes shared by the different knowledge areas. In this case the CONCEL description shown in the Figure 4 can be used as a generic vocabulary to be associated with each primary knowledge area.



.

Figure 8. A KSM generic knowledge model for traffic management. The *traffic management* knowledge is divided into (several) *local traffic management* areas and in a *conflict resolution* area. The *local traffic management* area is, in turn, divided into *identification* knowledge and *control proposals* knowledge.

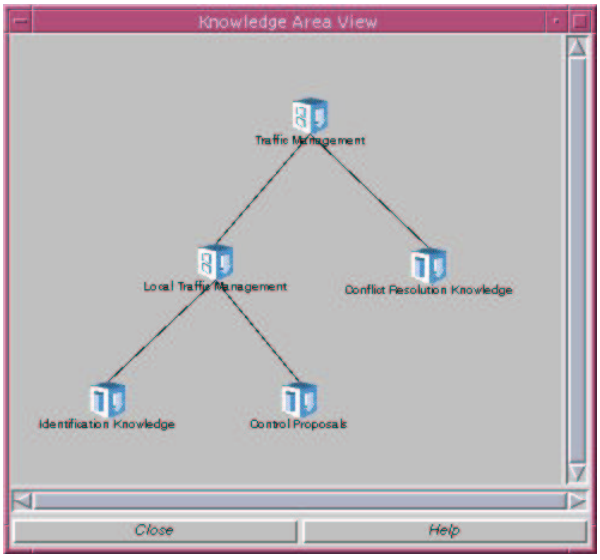


Figure 9. Functional model associated with the knowledge model of the Figure 8. The traffic management knowledge area has associated a *monitoring* task. The LINK method associated to this task is using both the *monitoring* tasks of *local traffic management* knowledge areas and the *conflict resolution* task carried out by using the *conflict resolution* knowledge. Each local monitoring is performed by means of the situation identification of the area and the control proposals generation for this area.

```
CONCEPT Frame IS A Concept.
ATTRIBUTES:
   Parent: INSTANCE OF Frame,
   Query Slots: INSTANCE OF Query Slot,
   Deduced Slots: INSTANCE OF Deduced Slot.

CONCEPT Slot IS A Concept.
ATTRIBUTES:
   Slot Attribute: ANY.

CONCEPT Query Slot IS A Slot.

CONCEPT Deduced Slot IS A Slot.
```

Figure 10. The abstract terminology associated to the *frame based* primitive of representation. *Frames* have associated *parent* frames, a set of *query* slots and a set of slots which can be deduced. The slots have associated *attributes*.

The third step in the system development is to select appropriate representation primitives to be associated to each primary area. In this case it is assumed that the *rule based* primitive is a suitable one both for the representation of the *identification* knowledge area and for the *conflict resolution* one. The abstract terminology associated to this primitive of representation has been already described in the Figure 5. For the representation of the *control proposal* knowledge a *frame based* primitive of representation is considered a good choice in order to map traffic situations to control states in the device. The abstract terminology managed by this primitive of representation is described in Figure 10. The subdivision between *query slots* and *deduced slots* is an ontological commitment, because the only task of this primitive, *matching*, accepts as input role a set of *query slots* and gives as output roles a set of *deduced slots*. There is a single *match* inference method associated with this task.

```
PRIMITIVE rules.
VOCABULARY IMPORTATION
Detector MAYBE A Precondition Concept.

      Detector.Intensity Measure MAYBE A Precondition Concept.Attribute.
      Detector.Occupancy Measure MAYBE A Precondition Concept.Attribute.
      Speed Detector.Speed Measure MAYBE A Precondition Concept.Attribute.

Traffic Situation MAYBE A Postcondition Concept.
      Traffic Situation.Presence MAYBE A Postcondition Concept.Attribute.

TASK ASSOCIATIONS

Situation Identification PERFORMED BY deduction(backward chaining)
INPUT Traffic state -> Initial facts,
      <Traffic Situation.Presence> -> Goals
OUTPUT Derived facts -> Traffic Situation.
```

Figure 11. Association of the *rule based* primitive to the *identification* knowledge area. The terminological importation is analogous to that shown in Figure 6. The *situation identification* task is carried out by the *backward chaining* method of the *deduction* task in the *rule based* primitive of representation.

```
PRIMITIVE rules.
VOCABULARY IMPORTATION

Control Device MAYBE A Rule Concept.

      Control Device.State MAYBE A Rule Concept.Attribute.

TASK ASSOCIATIONS

Conflict resolution PERFORMED BY deduction(forward chaining)
INPUT Control proposals -> Initial facts,
      <Control Device.State> -> Goals
OUTPUT Derived facts -> Fixes.
```

Figure 12. Association of the *rule based* primitive to the *conflict resolution* knowledge area. In this case the terminological importation specify that the allowed concepts in the rules are those associated with states of the control devices. The *conflict resolution* task is carried out by the forward *chaining* method of the *deduction* task in the *rule based* primitive of representation.

Once suitable primitives of representation, which are accomplishing the knowledge needs of each primary knowledge area, has been selected, they must be adapted to fit with these knowledge areas. This adaptation is performed by writing suitable LINK – S definitions for each primitive and primary area. Figure 11 shows the LINK – S definition binding the *rule based* primitive of representation with the *identification* knowledge area. Figure 12 shows the LINK – S definition that makes possible the same reusable knowledge representation component to be use for another different purpose (to formalise knowledge about conflict resolution in traffic control proposals). Finally, Figure 13 shows the LINK - S definition that adapts the *frame based* primitive for representing the local proposals associated with traffic control devices. The terminological importation associated with the *frame based* primitive of representation doesn't included any extension associated to *frame* (i.e, $\sigma^{-1}(\textit{frame}) = \perp$). The effects of a partial inverse of the functions associated within a terminological mapping must be defined with each primitive validation component. In this case it seems reasonable to think that the primitive is supposed to assume that no terminological constraints are imposed with respect to the names of the frames.

```
PRIMITIVE frames.
VOCABULARY IMPORTATION

Traffic situation MAYBE A Query Slot.
      Traffic situation.presence MAYBE A Query Slot.Slot Attribute.

Control device MAYBE A Deduced Slot.
      Control device.state MAYBE A Deduced Slot.Slot Attribute.

TASK ASSOCIATIONS

Proposal Generation PERFORMED BY matching(match)
INPUT Traffic situation -> query slots,
OUTPUT Deduced slots -> Proposal.
```

Figure 13. Association of the *frame based* primitive to the *control proposals* knowledge area. The query slots can be traffic situations, and the deductions are states of the control devices. The *proposal generation* task is carried out by match the traffic situation against the frame base.

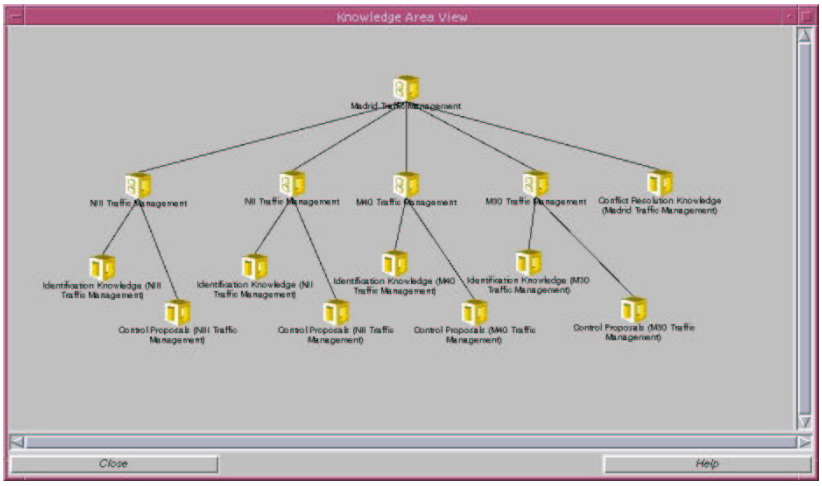


Figure 14. Specialisation of the generic model in Figure 8 in the city of Madrid. Four different local areas are identified. Each local area has assigned its local traffic management knowledge.

In this point the generic model can be specialised to meet the special needs of a given problem instance. It is performed instantiating the *traffic management* knowledge area over a given city and duplicating each *local management* knowledge area to cover the different control zones of this city. For each basic area the generic traffic terminology of Figure 4 is specialised in terms of the so – called *domain conceptual vocabularies*. Furthermore, the different domain vocabularies related to each area would be also associated to the *conflict resolution* knowledge. Figure 14 shows the knowledge area topology associated to a hypothetical instantiation of the model in Madrid. Figure 15 shows a fragment of a domain terminological specification associated with an area. In this situation, the specific knowledge associated with the case can be acquired by using the acquisition facilities of each primitive. These acquisition facilities will be *sensible* to the terminological importation performed at the generic level. When the case knowledge has been formalised, the system is right to be used.

```
CONCEPT M30_IN_PK200 INSTANCE OF SECTION.
ATTRIBUTES:
   Next Sections = M30_IN_PK600,
   Traffic Detector = DIN200.

CONCEPT M30_IN_PK600 INSTANCE OF SECTION.
ATTRIBUTES:
   Previous Sections = M30_IN_PK200.
   Next Sections: INSTANCE OF Section = M30_IN_PK800,
   Traffic Detector: DIN600.

      ....

CONCEPT DIN200 INSTANCE OF Speed Detector.

CONCEPT DIN600 INSTANCE OF Detector.

      ....
```

Figure 15. Specific traffic terminology in the highway M30 located in the city of Madrid.

8. Related work

The idea of terminological importation can be related to the more general framework of *ontological mappings* [Park et al, 98]. These mappings are used in different knowledge modelling frameworks to relate ontologies formulated to different levels [Wielinga et al, 92; Puerta et al, 92; Motta, 97]. In particular, the idea of ontological mappings is used in modelling languages as OCML [Motta, 98], KARL [Fensel, 95] or ML^2 [van Harmelen,Balder,92], or in environments such as PROTEGE-II [Gennari et al, 94]. The main difference between terminological importation and the mappings used in these knowledge – modelling approaches is that terminological importation is partial in nature. Knowledge in primitives of representation is not derived as a refinement of CONCEL definitions. Indeed, CONCEL can describe only terminological knowledge. So, terminological importation can be seen as a very restricted kind of ontological mapping, yet enough powerful to specify a concretion of the abstract terminologies associated to primitives of representation in terms of domain terminologies. This approach conceives the terminological importation as an *instantiation* of an abstract terminology, or as a *population* of the abstract terminology with a set of domain concepts. In addition, the simplicity of the terminological importation drives to simple and efficient implementations, which is of crucial importance in the final implementations of knowledge – based systems.

9. Conclusions and future work

This paper illustrates the use of terminological importation as a suitable approach to adapt reusable knowledge representation components to specific domains. By specifying those domain concepts which can fill a given linguistic category in a representation formalism domain - sensible copies of a given domain independent component can be obtained. This is a necessary condition to maximise the usability of a reusable, domain - independent knowledge representation components. To reuse a component across different domains, the domain - dependent ontological commitments would be excluded from the structure of the component. However, this fact drives to an undesired generality in the class of sentences that can be described in the representation formalism provided by the component. This generality can be reduced with a mechanism to import concepts from the specific domain to the terminology of the component. The mechanism, for instance, produces a component specialised in the description of rules for detecting traffic situations from a generic rule based component, or a component to describe suitable signal plans for traffic devices from a frame based one. These adapted components are more usable than their generic counterparts, because the knowledge acquisition facilities are sensible to this terminological importation. In addition, the use of terminological importation fixes necessary conditions in order to ensure component interoperability: it is difficult to ensure the interoperability of components if these disagree in the basic common vocabulary. This agreement can be reached by importing concepts from shared conceptual vocabularies.

Some considerations related with the introduced adaptation scheme require additional future work. An important matter is how to cope with heterogeneous local explanations provided by inference methods in a domain - oriented way, and how combine them in order to elaborate a global explanation of the results

obtained in the execution of the operational model. The use of abstract states associated to the problem solving methods (which can be described in terms of the local abstract terminologies) seems to be a useful approach to be explored. A more in - depth external characterisation of the components (description of the structure of input and output task roles in terms of the abstract terminology, description of the services provided by the knowledge acquisition module, etc) together with a more flexible interoperability theory between components would be also provided. Currently this interoperability is achieved by assembling components by means of the LINK language. However more flexible assembling policies could be studied, such as the use of control components to assembly other components. The abstract terminology of such components would be formulated in terms of the services provided by the potential components to be assembled. This mechanisms would drive to reusable control knowledge representation components that could be adapted by mean of an *importation of services* in order to operate in concrete knowledge - based architectures.

10. References

[Cuena,Molina,94] Cuena J., Molina,M.: "KSM: An Environment for Knowledge Oriented Design of Applications Using Structured Knowledge Architectures" in "Applications and Impacts. Information Processing'94", Volume 2. K. Brunnstein and E. Raubold (eds.) Elsevier Science B.V. (North-Holland), IFIP. 1994.

[Cuena et al, 96] Cuena J. Hernández J. Molina M.: "Knowledge Oriented Design of an Application for Real Time Traffic Management: The TRyS System". Proc. 12th European Conference on Artificial Intelligence ECAI 96. Budapest, Hungary, 1996.

[Cuena,Molina,97] Cuena J.,Molina, M.: "KSM: An Environment for Design of Structured Knowledge Models" in "Knowledge-based Systems: Advanced Concepts, Techniques and Applications". S.G. Tzafestas (Ed.) Publisher World Scientific Publishing Company. 1997

[Fensel,95] Fensel, D.: "The Knowledge Acquisition and Representation Language KARL". Kluwer Academic Publisher. Boston. 1995.

[Gennari et al, 94] Gennari J.H. Tu S. Rothenfluh, T. Musen, M.A.: "Mapping Domains to Methods in Support of Reuse". Tech. Report KSL-93-67. Knowledge Systems Laboratory. Stanford University. 1994.

[Molina et al.,97] Molina M. Gómez A. Sierra J.L: "Reusable Components for Building Conventional and Knowledge-based Systems: The KSM approach". The 9th International Conference on Software Engineering and Knowledge Engineering SEKE 97. Madrid, Spain, 1997.

[Molina et al.,98] Molina M.,Sierra J.L.Serrano J.M.: "A Language to Formalize and to Operationalize Problem Solving Strategies of Structured Knowledge Models". 8th Workshop on Knowledge Engineering: Methods & Languages KEML 98. Karlsruhe, Germany, 1998

[Motta,97] Motta,E.: "Reusable Components for Knowledge Modelling". Ph.D.Thesis. Knowledge Media Institute.The Open University. UK. 1997.

[Motta,98] Motta,E.: "An Overview of the OCML Modelling Language". 8th Workshop on Knowledge Engineering: Methods & Languages KEML 98. Karlsruhe, Germany, 1998

[Puerta et al,92] Puerta A. Egar J.W. Tu S. Musen M.A.: "A Multiple-Method Knowledge Acquisition Shell for the Automatic Generation of Knowledge Acquisition Tools". Knowledge Acquisition, 4(2). Pp. 171-196. 1992.

[Park et al,98] Park J.Y. Gennari J.H. Musen M.A.: "Mappings for Reuse in Knowledge-based Systems". 11th Workshop on Knowledge Acquisition, Modelling and Management KAW 98. Banff, Canada, 1998.

[vanHarmelen,Balder,92] Van Harmelen F. Balder J.R.: "$(ML)^2$: A Formal language for KARL models of expertise". Knowledge Acquisition, 4(1). Pp 127-161. 1992.

[Wielinga et al, 92] Wielinga B.J., Schereiber A.T. Breuker J.: "KADS: A Modelling Approach to Knowledge Engineering". Knowledge Acquisition 4(1). Pp. 5-53. 1992.